iPhone View Controller Techniques

New features and best practices

Session 123

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

iPhone View Controller Techniques

New features and best practices

Evan Doll
iPhone Software Engineer

Goals for Today

- Adopt **cool and useful features** in your new or existing apps
- **Don’t waste time** reinventing the wheel
- Make your app **future-ready**

What We'll Cover

- Why View Controllers?
- New features in iPhone OS 3.0
- Best practices

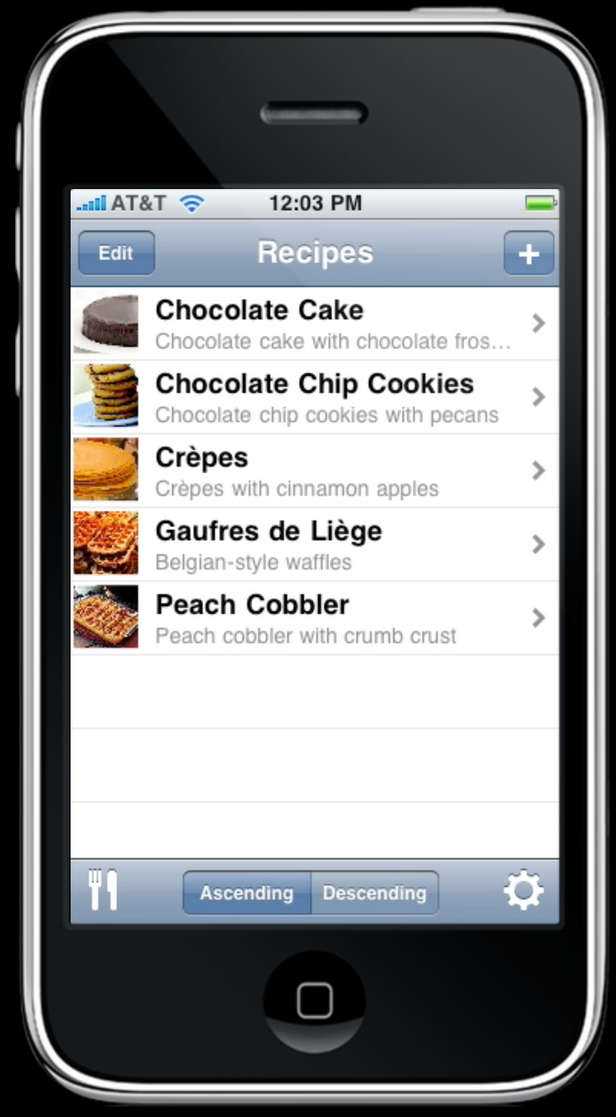
Why View Controllers?

Designing iPhone Apps

- Apps made up of multiple screens
- Each screen does one thing well
- Connected using common app flows

Designing iPhone Apps

- Apps made up of multiple screens
- Each screen does one thing well
- Connected using common app flows



The Role of Controllers in Your App

- Connect model and view
  - Populate views with data
- Provide application logic
  - Respond to actions and events

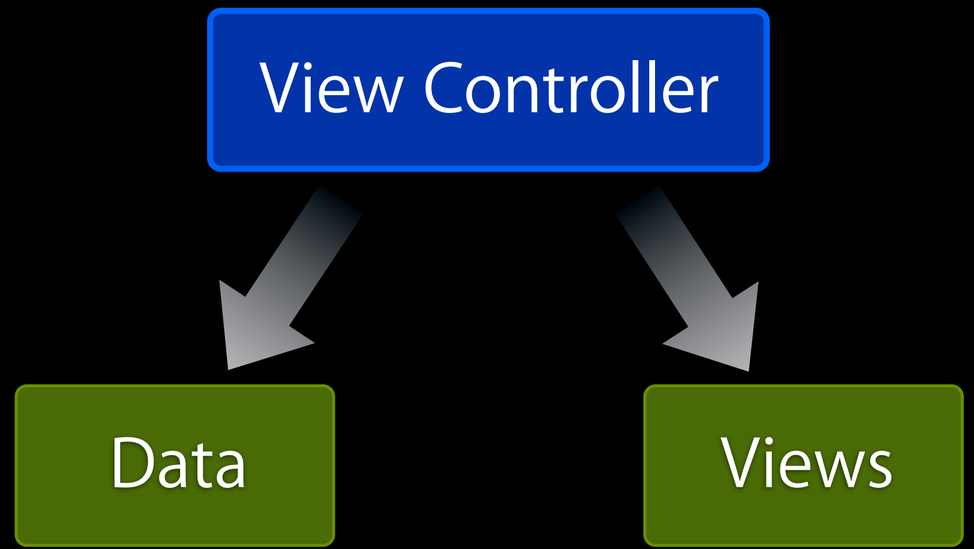
The Brains of Your App

A Starting Point for Your Controllers

- Include some basic functionality
  - Loading a view from a NIB
  - Respond to appearing and disappearing
  - Presenting modal content
  - Interface rotation

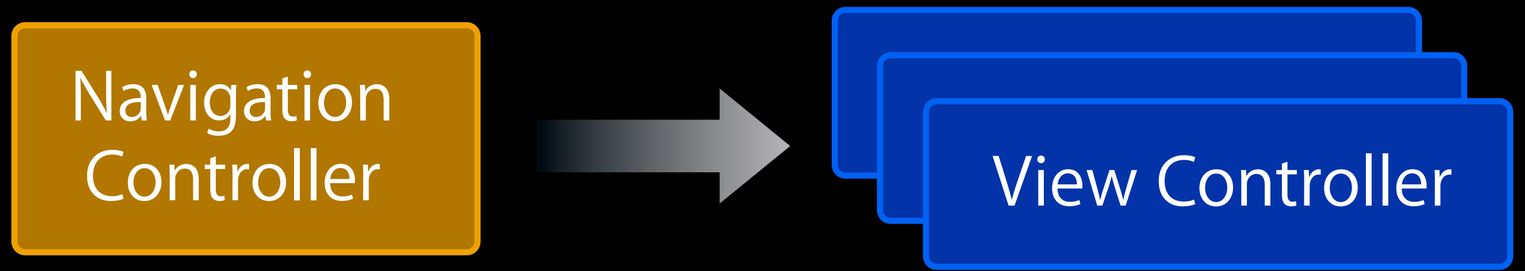
UIViewController

- Basic building block
- Manages a screenful of content
- Create a subclass, add application logic



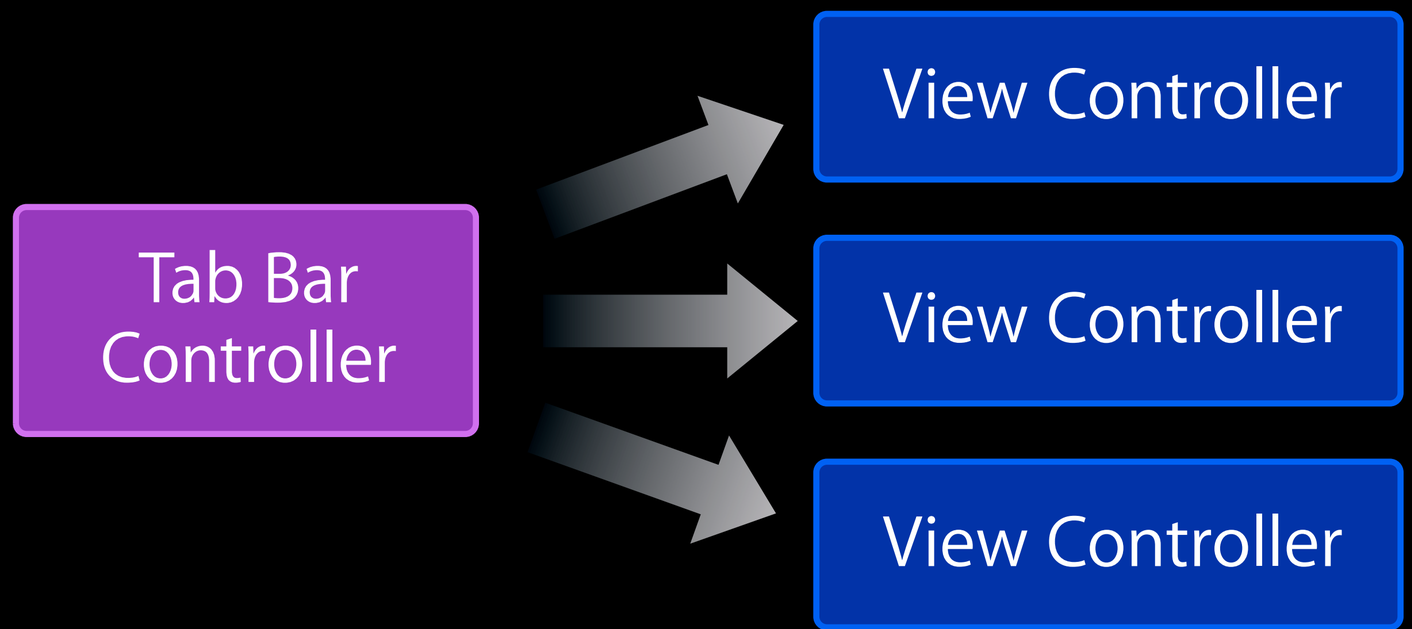
Connecting View Controllers

- Create your view controllers, one for each screen
- Use container view controller to **plug them together**



Connecting View Controllers

- Create your view controllers, one for each screen
- Use container view controller to **plug them together**



Using a Navigation Controller

- Drill down into a hierarchy of data
- Push and pop as your user navigates
- Navigation bar updates automatically

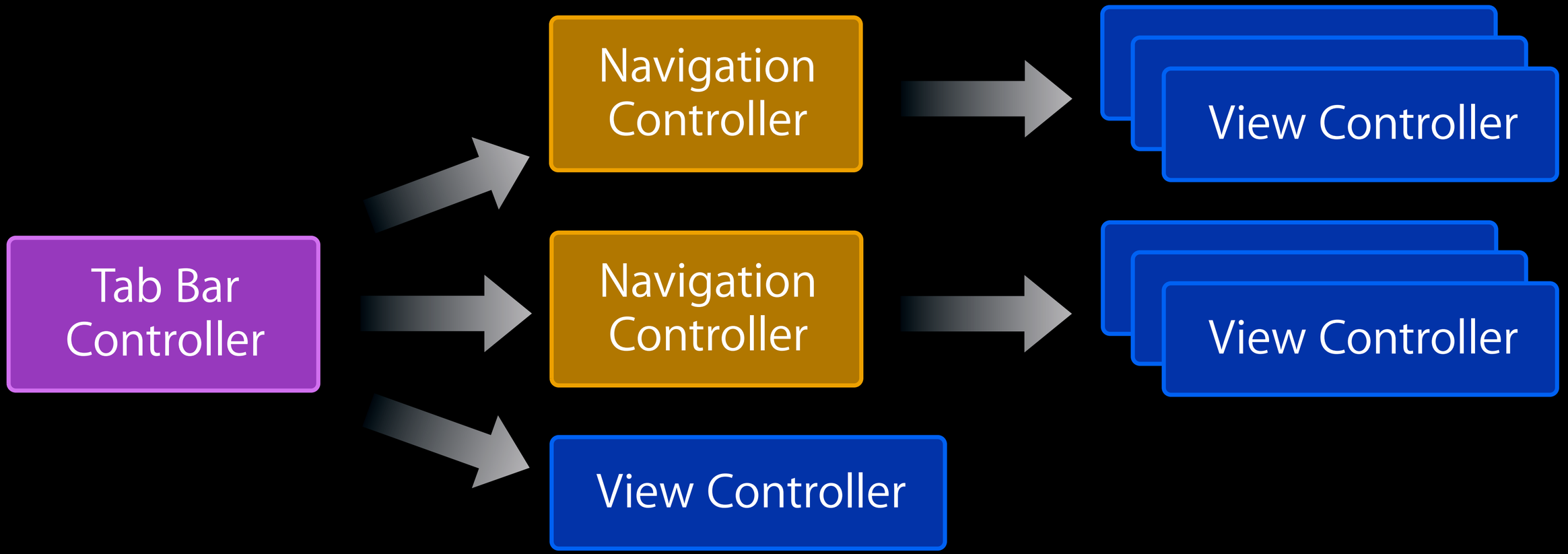
```
// In response to user action
[navigationController pushViewController:viewController
                                animated:YES];
```

Using a Tab Bar Controller

- Separate, self-contained modes
- User taps tab bar to switch

```
// Set up the tab bar controller
NSArray *myViewControllers = ...;
tabBarController.viewControllers = myViewControllers;
```

Combining Tab Bar + Navigation



Presenting Modal Content

- For adding or picking data
- Covers entire screen

Ready to Present View Controllers

- Image picker
- People picker
- Mail compose
- Media picker

More on View Controllers…

- WWDC 2008 on iTunes
  - "Understanding View Controllers" and "Mastering View Controllers"
  - http://developer.apple.com/adconitunes/
- iPhone Application Programming at Stanford
  - http://cs193p.stanford.edu

iPhone View Controllers Lab	iPhone Lab A Friday 9:00AM

New Features in iPhone OS 3.0

New Features in iPhone OS 3.0

- Contextual toolbars
- Flipping your app
- Mixing interface orientations
- Alternate landscape UI
- Full screen layout

Contextual Toolbars

Contextual Toolbars

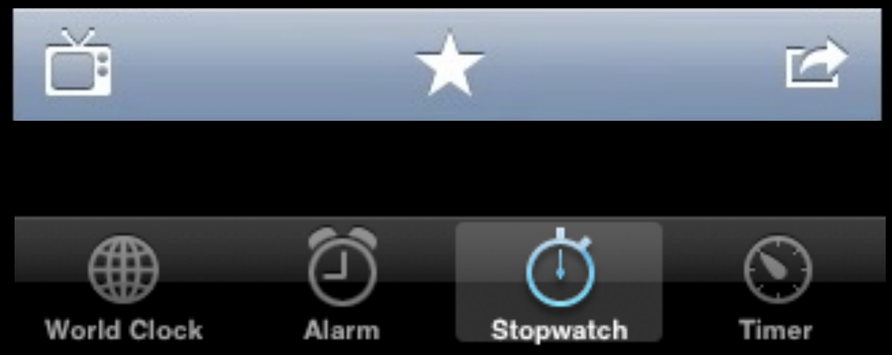
With a navigation controller

Contextual Toolbars

- Navigation controller may display a toolbar
- Each view controller **has its own toolbar items**
- When navigation occurs, **toolbar automatically updates**
- The hidesBottomBarWhenPushed property works for toolbars

Toolbars vs. Tab Bars

- Toolbars are for performing **actions**
  - **Momentary** highlighting
- Tab bars are for indicating **modes**
  - **Persistent** highlighting



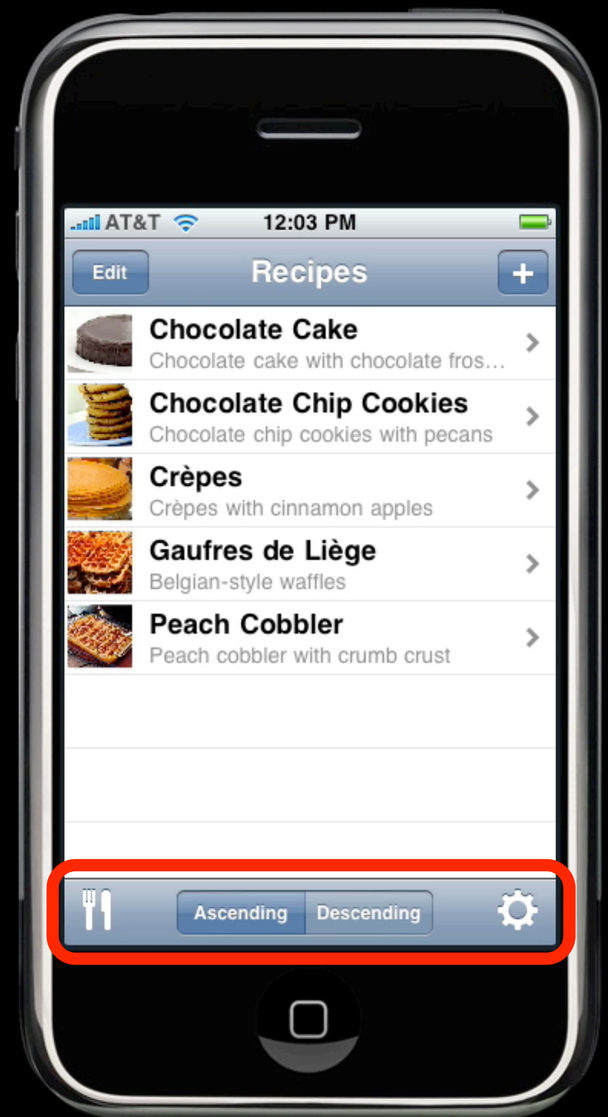
- Your application may use one or the other, but **not both**

Using Contextual Toolbars

```
navigationController.toolbarHidden = NO;
```

```
listViewController.toolbarItems = listItems;
```

```
detailViewController.toolbarItems = detailItems;
[nav pushViewController:detailViewController
                animated:YES];
```



What Did We Just See?

- View controllers have toolbar items
- Navigation controller automatically updates the toolbar

Flipping Your App

Flipping Your App

Using modal transition styles

Modal Transition Styles

```
typedef enum {
   UIModalTransitionStyleCoverVertical,
   UIModalTransitionStyleFlipHorizontal,
   UIModalTransitionStyleCrossDissolve
} UIModalTransitionStyle;
```

```
// UIViewController
@property(nonatomic,assign)
UIModalTransitionStyle modalTransitionStyle;
```

Using Modal Transition Styles

- Set a modal transition style
- Present and dismiss the view controller

Using Modal Transition Styles

```
backViewController.modalTransitionStyle =
    UIModalTransitionStyleFlipHorizontal;
```

```
[frontViewController
    presentModalViewController:backViewController
                      animated:YES];
```

```
[frontViewController
    dismissModalViewControllerAnimated:YES];
```

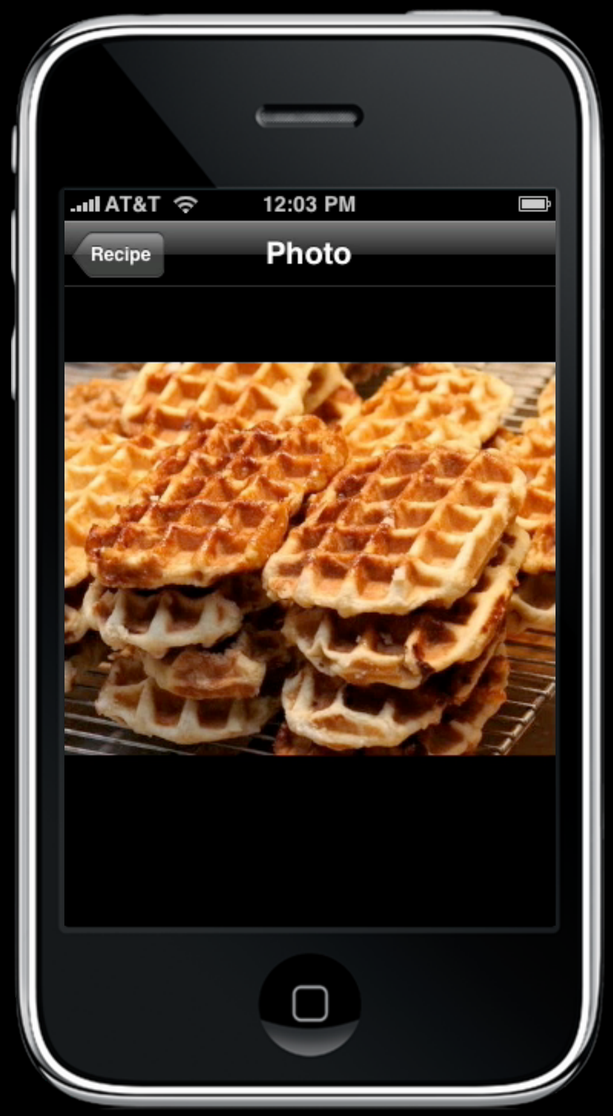
What Did We Just See?

- Flipping view controllers used to be nontrivial
- Now it takes two lines of code

Mixing Interface Orientations

Mixing Interface Orientations

By popular request



Mixing Interface Orientations

- Mixed orientations are now supported in two situations
  - Presenting modally
  - Within a navigation stack
- Implement -shouldAutorotateToInterfaceOrientation:
  - Return YES for **supported orientations**
  - **Simple** implementation
  - Design time, not runtime logic

Implementing -shouldAutorotate…

- Portrait-only view controller

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
{
    return interfaceOrientation == UIInterfaceOrientationPortrait;
}
```

- Landscape view controller

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
{
    return UIInterfaceOrientationIsLandscape(interfaceOrientation);
}
```

Presenting Modally with Mixed Orientations

- Recipe list view controller
  - Supports **portrait orientation only**
- Utensils view controller
  - Supports **landscape orientations**

Navigating with Mixed Orientations

- Photo view controller
  - Supports **portrait and landscape orientations**
- Previous view controller in navigation stack
  - Supports **portrait orientation only**

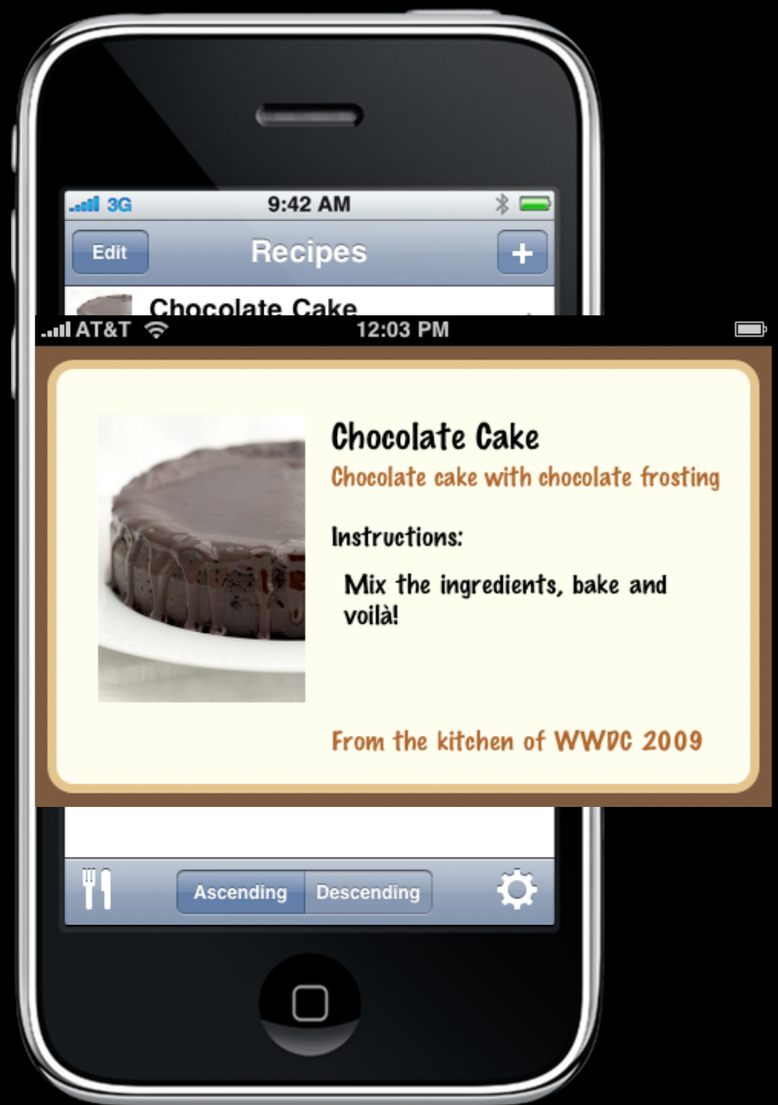
What Did We Just See?

- No new API required for mixing interface orientations
- **Indicate supported orientations** via -shouldAutorotate…

Alternate Landscape UI

Alternate Landscape UI

Crossfade to a different orientation

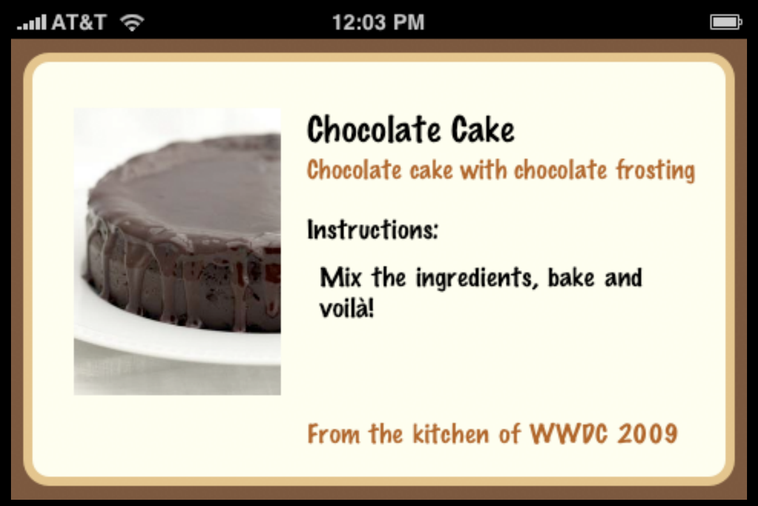


Alternate Landscape UI

- Use **two view controllers**
  - Portrait view controller
  - Landscape view controller
- Observe UIDevice orientation change notifications
  - Present when rotated to landscape
  - Dismiss when rotated to portrait
- Use UIModalTransitionStyleCrossDissolve

Adding an Alternate Landscape UI

Recipe card in landscape




```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear];
    [application setStatusBarStyle:UIStatusBarStyleDefault animated:animated];
}
```

Adding an Alternate Landscape UI

Observe UIDevice rotation

```
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    ...

    [notificationCenter addObserver:self
                           selector:@selector(deviceOrientationDidChange:)
                               name:UIDeviceOrientationDidChangeNotification
                             object:nil];
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (UIDeviceOrientationIsLandscape(orientation) &&
        [navigationController.modalViewController == nil &&
        ![self canTopViewControllerRotateToLandscape])
    {
        cardViewController = ...;
        cardViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
        [navigationController presentModalViewController:cardViewController
                                                animated:YES];
    } else if (...) {
        ...
    }
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (UIDeviceOrientationIsLandscape(orientation) &&
        [navigationController.modalViewController == nil &&
        ![self canTopViewControllerRotateToLandscape])
    {
        cardViewController = ...;
        cardViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
        [navigationController presentModalViewController:cardViewController
                                                animated:YES];
    } else if (...) {
        ...
    }
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (UIDeviceOrientationIsLandscape(orientation) &&
        [navigationController.modalViewController == nil &&
        ![self canTopViewControllerRotateToLandscape])
    {
        cardViewController = ...;
        cardViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
        [navigationController presentModalViewController:cardViewController
                                                animated:YES];
    } else if (...) {
        ...
    }
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (UIDeviceOrientationIsLandscape(orientation) &&
        [navigationController.modalViewController == nil &&
        ![self canTopViewControllerRotateToLandscape])
    {
        cardViewController = ...;
        cardViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
        [navigationController presentModalViewController:cardViewController
                                                animated:YES];
    } else if (...) {
        ...
    }
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (UIDeviceOrientationIsLandscape(orientation) &&
        [navigationController.modalViewController == nil &&
        ![self canTopViewControllerRotateToLandscape])
    {
        cardViewController = ...;
        cardViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
        [navigationController presentModalViewController:cardViewController
                                                animated:YES];
    } else if (...) {
        ...
    }
}
```

Adding an Alternate Landscape UI

Cross dissolve to landscape

Adding an Alternate Landscape UI

Return to portrait

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    ...
    } else if (orientation == UIDeviceOrientationPortrait &&
               navigationController.modalViewController == cardViewController)
    {
        [navigationController dismissModalViewControllerAnimated:YES];
        [cardViewController release];
        cardViewController = nil;
    }
}
```

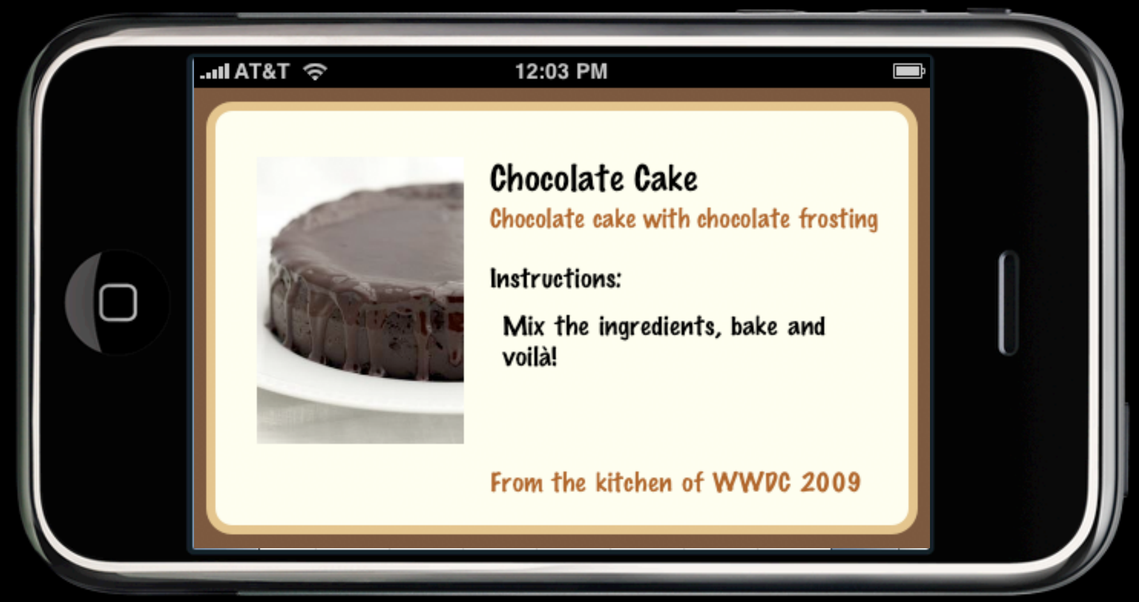
Adding an Alternate Landscape UI

Return to portrait

```
- (void)deviceOrientationDidChange:(NSNotification *)notification
{
    ...
    } else if (orientation == UIDeviceOrientationPortrait &&
               navigationController.modalViewController == cardViewController)
    {
        [navigationController dismissModalViewControllerAnimated:YES];
        [cardViewController release];
        cardViewController = nil;
    }
}
```

Adding an Alternate Landscape UI

Return to portrait



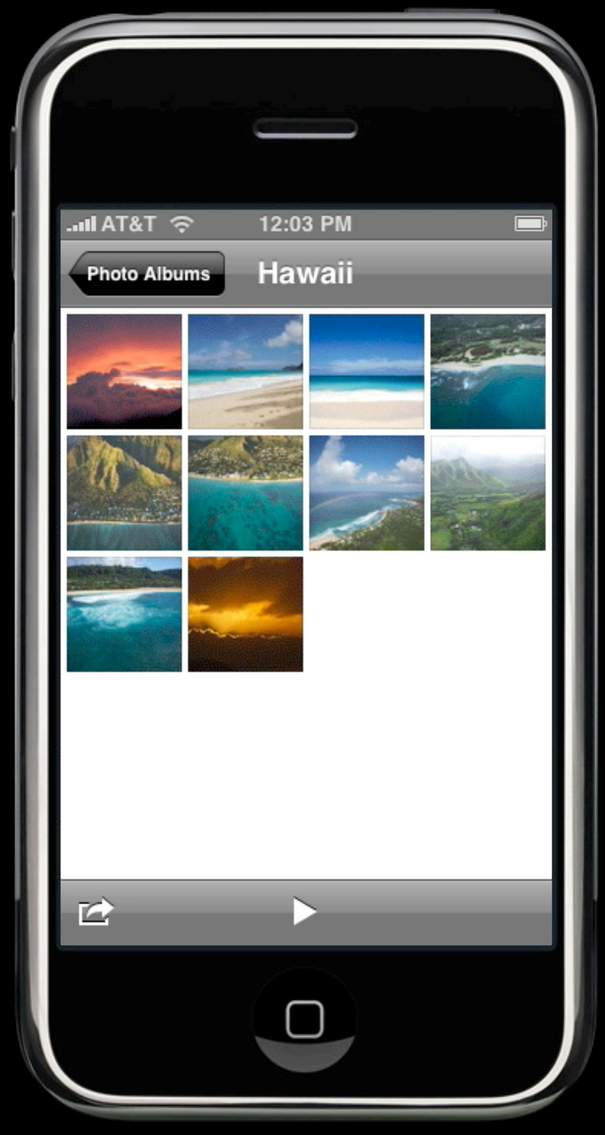
What Did We Just See?

- Two view controllers
- Present and dismiss modally in response to device rotation
- Use UIModalTransitionStyleCrossDissolve

Full Screen Layout

Full Screen Layout

Overlapping content with the status bar



Overlapping Navigation Bar

- Make navigation bar **translucent** to automatically overlap content
  - Has always worked for scroll views
  - Extended to all views for iPhone OS 2.2

Overlapping Status Bar

- Window and containing view controller sized to full screen
- View controller **wantsFullScreenLayout** set to YES
- Update status bar style yourself

Using Full Screen Layout

Positioning the navigation controller

```
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    CGRect fullScreenFrame = [[UIScreen mainScreen] bounds];
    UIWindow *window = [[UIWindow alloc] initWithFrame:fullScreenFrame];

    UINavigationController *navigationController = ...
    // No longer needs to be [[UIScreen mainScreen] applicationFrame]!
    navigationController.view.frame = window.bounds;

    [window makeKeyAndVisible];
}
```

Using Full Screen Layout

Positioning the navigation controller

```
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    CGRect fullScreenFrame = [[UIScreen mainScreen] bounds];
    UIWindow *window = [[UIWindow alloc] initWithFrame:fullScreenFrame];

    UINavigationController *navigationController = ...
    // No longer needs to be [[UIScreen mainScreen] applicationFrame]!
    navigationController.view.frame = window.bounds;

    [window makeKeyAndVisible];
}
```

Using Full Screen Layout

Positioning the navigation controller

```
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    CGRect fullScreenFrame = [[UIScreen mainScreen] bounds];
    UIWindow *window = [[UIWindow alloc] initWithFrame:fullScreenFrame];

    UINavigationController *navigationController = ...
    // No longer needs to be [[UIScreen mainScreen] applicationFrame]!
    navigationController.view.frame = window.bounds;

    [window makeKeyAndVisible];
}
```

Using Full Screen Layout

Indicating your layout preference

```
photoViewController = ...;
photoViewController.wantsFullScreenLayout = YES;
[navigationController pushViewController:photoViewController
                                animated:YES];
```

Using Full Screen Layout

Updating bar styles

```
// When pushing...
[application setStatusBarStyle:UIStatusBarStyleBlackTranslucent animated:YES];
navigationController.navigationBar.barStyle = UIBarStyleBlack;
navigationController.navigationBar.translucent = YES;
```

```
// When popping...
[application setStatusBarStyle:UIStatusBarStyleDefault animated:YES];
navigationController.navigationBar.barStyle = UIBarStyleDefault;
navigationController.navigationBar.translucent = NO;
```

Using Full Screen Layout

Putting it all together

What Did We Just See?

- Ensure that window and containing view controllers are full-size
- Set wantsFullScreenLayout to YES
- Use translucent navigation bar and status bar
- Restore previous bar styles on disappear

Best Practices

Respecting the View Hierarchy

- Internal view hierarchies are **always subject to change**
  - Navigation bar
  - Navigation and tab bar controllers
  - Other view controllers (camera, address book)
- **Unsafe** to depend on undocumented details of system views
  - Manipulating subviews
  - Adding your own subviews

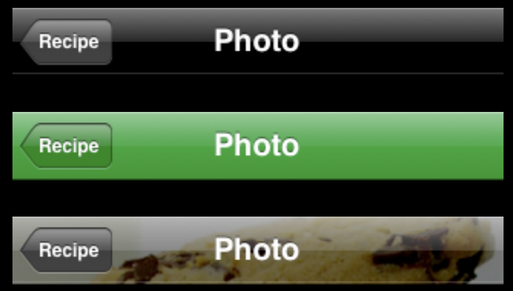
- Work with system frameworks, not against them

Container View Controllers

- Some view controllers are designed to contain other view controllers
  - Navigation and tab bar controllers
- **Not recommended** to create your own container view controller
- To flip or crossfade view controllers, **use modal transition styles**

Navigation Bar Appearance

- Navigation controllers own their navigation bars
- Okay for you to customize **certain properties**
  - Bar style
  - Tint color
  - Translucent



- Other properties are **not safe to customize**
  - Frame
  - Hidden

Showing and Hiding the Navigation Bar

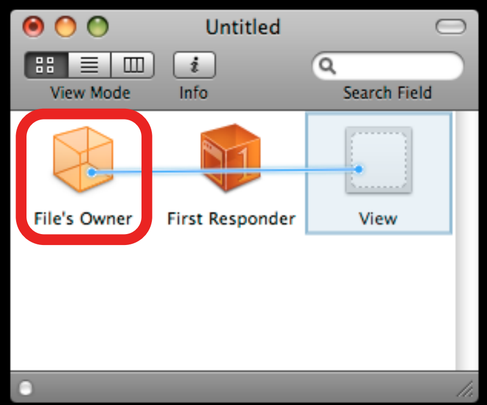
- Use -setNavigationBarHidden:animated:
- Animate horizontally during push or pop
  - Call from -viewWillAppear: or -viewWillDisappear:

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];

    [self.navigationController
        setNavigationBarHidden:YES animated:animated];
}
```

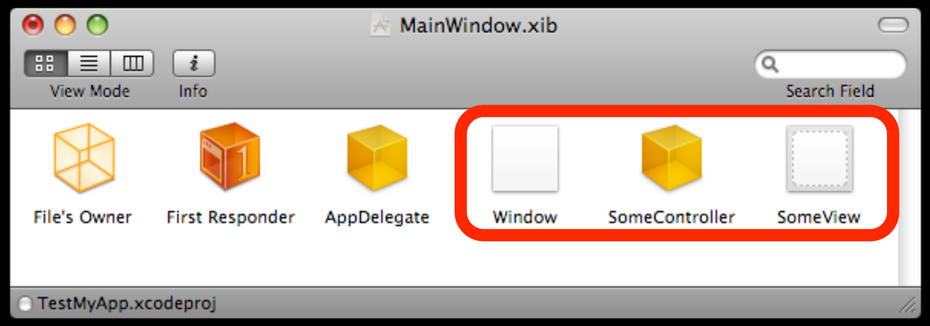
When to Use Interface Builder?

- Almost always recommended!
- Especially useful when creating a view with many subviews
- One view controller per NIB
  - Make connections to view controller via File's Owner



Nibs and Memory Management

- Top-level nib objects are **autoreleased**
  - Retain to keep around after loading



- An IBOutlet without a setter will be **retained by default**
  - Release it in -dealloc!
  - For better clarity, declare a property
- Release and nil your IBOutlets in -viewDidUnload

Summary

- Build your application with view controllers
- Adopt cool and useful features using new API
- Follow best practices in your app

More Information

Matt Drance
Application Frameworks Evangelist
mdrance@apple.com

Documentation
iPhone Dev Center
http://developer.apple.com/iphone

iPhone View Controllers Lab	iPhone Lab A Friday 9:00AM

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09